A new algorithm for Automatic Word Classification based on an Improved Simulated Annealing Technique

Kamel Smaïli, François Charpillet, Jean-Paul Haton

A New Algorithm for Automatic Word Classification based on an Improved Simulated Annealing Technique

K. Smaïli, F. Charpillet, J-P. Haton
CRIN /INRIA Lorraine BP 239 54506 Vandœuvre lès-Nancy
smaili@loria.fr (Tel 33 83-59-20-83, Fax 33 83-41-30-79)

Abstract

In this work, we present a new method for clustering words into equivalence classes within a bigram word (which is extensible to a trigram word). This method is based upon an improved approach of the simulated annealing technique. The main idea consists in defining a transition function instead of a random transition function as usually used in classical approaches. Our transition function specifies the way the classes may evolve during the simulating annealing process. It is based on a set of stochastic rules based on linguistic criteria which select the set of words that can be moved from one class to another. This method allows to palliate the main drawbacks of the classical approach: time complexity and random transition function.

In this paper, we compare the performances of both approaches: the classical simulated annealing and the proposed one and we report the results of an evaluation on two french databases. The classical approach is 2,5 times slower than our approach and despite of a low value of perplexity the classification obtained is not as good as the one given by our approach.

I Introduction

Currently, nearly all language models in speech recognition systems are probabilistic. This choice is due to the difficulty of writing a computational explicit formal grammar for natural language. That is why the probabilistic approach has became a very popular technique in the last few years. The basic idea in this approach is to learn probabilities from a large corpora in order to predict future behaviour of the model. Let w be a word occuring n times in the training corpora. If w can be tagged k times by the syntactic category C, then we assume that the relative frequency of the word w is given by: $P(C|w) = \frac{k}{n}$. If this probability is computed from a large corpora of N events, it will be used to guide our prediction with the $N+1$st case. This simple ratio is called the *maximum likelihood estimator* (MLE).

More generally, we can formulate the problem of learning a language model as: given $w_1, w_2, \ldots w_n$ a sequence of words constituting all the sentences of the corpora, how to determine the syntactic categories $C_1, C_2, \ldots C_n$ which maximize: $P(C_1 C_2 \ldots C_n | w_1 w_2 \ldots w_n)$. The answer is given by using the Bayes rule.

$$P(C_1 C_2 \ldots C_n | w_1 w_2 \ldots w_n) = \frac{P(C_1 C_2 \ldots C_n) \times P(w_1 w_2 \ldots w_n | C_1 C_2 \ldots C_n)}{P(w_1 w_2 \ldots w_n)} \quad (1)$$

As we are interested in finding $C_1 \ldots C_n$ that gives the maximum value to the left term of the previous equation, the denominator of the formula (1) does not affect the optimization. So the problem is reduced to find $C_1 \ldots C_n$ that maximizes Q.

$$Q = P(C_1 C_2 \ldots C_n) \times P(w_1 w_2 \ldots w_n | C_1 C_2 \ldots C_n) \quad (2)$$

with

$$P(C_1 C_2 \ldots C_n) = \prod_{i=1}^{n} \hat{P}(C_i | C_1 \ldots C_{i-1}) \quad (3)$$

Unfortunately, the previous formula would take far too much data to generate reasonable

estimates, so we have to approximate this expression by the following one:

$$P(C_1 C_2 \ldots C_n) = \prod_{i=1}^{n} P(C_i | C_{i-k+1} \ldots C_{i-1}) \quad (4)$$

Equation (4) means that the probability of the sequence $C_1 \ldots C_n$ is approximated by the conditional probability of one class given the *k-1* previous categories. This model is called the *k-gram* model. In practice, *k* may be set to 2 or 3.

Regarding, the second term of the formula (2), we assume (to approximate it) that a word appears in a class independently of the preceding classes. Thus the second expression of the formula (2) is approximated by the product of the probability that each word occurs in the indicated class. Then equation (2) becomes :

$$P(w_1 w_2 \ldots w_n | C_1 C_2 \ldots C_n) = \prod_{i=1}^{n} P(w_i | C_i) \quad (5)$$

When *k* is restricted to 2 the formula (2) is computed as follows:

$$Q = \prod_{i=1}^{n} P(w_i | C_i) \times P(C_i | C_{i-1}) \quad (6)$$

To estimate the quantity Q, we use an imaginary class or category C_0. In our experiments, we assumed that each sentence is preceded by a period. In the following we will use the formula (6).

II The Necessity of Tagging

Given the formula (6), we can easily understand that speech or text has to be tagged. As a matter of fact, in this formula, the probability of a word depends on the class it belongs to and on its syntactic previous class. So, in order to estimate this probability, we need to tag each word of the training corpora. To do that, the dictionary of the application will contain an information about the category of each entry. In other words, that means that an operation of splitting the dictionary has to be done. This operation is known by linguists as the classification operation.

Let us give the definition of the classification: *The classification is the operation of splitting the dictionary into classes. Each class contains the words which have similar syntactic or semantic behaviours*. The splitting operation is not necessarly a mathematical partition, i.e., a word could be in several classes. For instance, a classification of verbs made by Gross produced about 2000 classes for 3000 verbs [Gross 75]. The operation of classification uses some features to build up classes, these features could be seen as contexts where the words will appear. The number of classes increases with the number of features. In a speech recognition system we do not need such a large and detailed classification; otherwise if the number of classes is very high, we will have the same problem of word estimation probabilities. There are two possibilities to make the classification: a hand classification and an automatic classification.

II.1 A Hand Classification

Several years ago, we developed a hand classification for French words for a prototype of a dictation system (MAUD) [Smaili 91]. We refined the eight elementary French grammatical classes, and we obtained 201 syntactic-semantic classes. The theoretical rule used to classify the dictionary is the following: *A word belongs to a class if any other word of the same class could be substituted to it in a context without any change in the syntactic structure of the sentence.*

To make that possible, we defined some syntactic features (contexts) and tried all the words of the dictionary on those contexts. The words which can appear in the same contexts are

arranged in the same class. For instance, the syntactic structure of checking words look likes :

NP V ☐ PREP NP

Where:

NP	is a noun phrase
V	is a verb
PREP	is a preposition
☐	the place of the word under study

The classification has been done on a dictionary of 37 000 entries. From the eighth grammatical classes of BDLEX (a lexical database for spoken and written French). We split each class into two classes: an opened class and a closed one.

The opened class is made up of the words which could be formed from the root of the word such as some verbs, adjectives, etc.

The closed class is made up of a finite number of words such as articles. This kind of classes could also contain words which are difficult to classify and those who have a specific function in the natural language such as the tool words of the natural language.

By applying the definition above, we built up 201 syntactic classes which are shared out as follows:

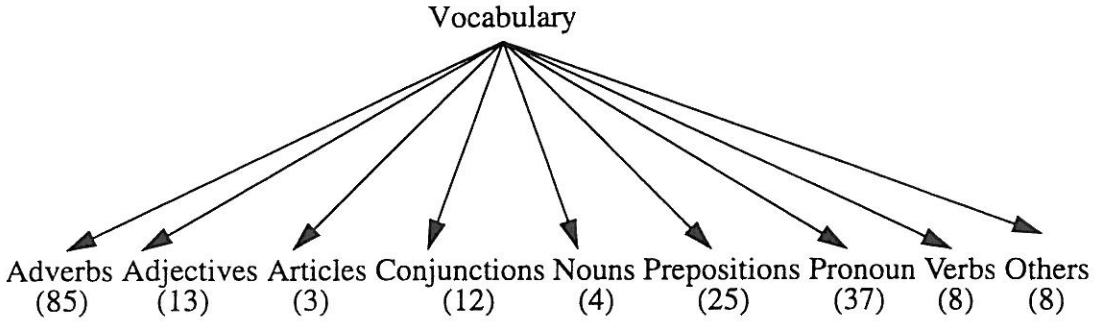

Fig. 1 The distribution of the word vocabulary on eight main categories

In figure 1, we give the number of sub-categories (classes) for the main grammatical categories. Note that, there is a grammatical class called *Others*. This class contains all the classes which are particular, e.g., which contains words which have more than four grammatical roles.

II.2 The Construction of the Language Model

With this classification, we built up a language model which is based on the formula (6). The values of the parameters in this formula are computed from a learning corpora. The problem of this formula is that if a word in the test phase has not been seen in the learning phase, the formula systematically gives a null probability. We handle this problem of sparse data by smoothing techniques. This technique consists in using biclasses and uniclass in the formula (6). In our experiments we used the following formula:

$$Q = \prod_{i=1}^{n} \left(P\left(C_i|w_i\right) + \delta \right) \times \left(\lambda_1 \cdot P\left(C_i\right) + \lambda_2 \cdot P\left(C_i|C_{i-1}\right) + \lambda_3 \cdot P\left(C_i|C_{i-1}C_{i-2}\right) + \lambda_4 \right) \quad (7)$$

with $\lambda_3 \gg \lambda_2 \gg \lambda_1$ and $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$

The weights which we used in our model, allow to give more importance to the model which has a wide context (that is why λ_3 is higher than λ_2 and λ_2 is higher than λ_1). $\lambda 4$ is a parameter which guarantees non-zero probability. As a matter of fact, if we do not know to which

class an unknown word belongs, the second term of the formula 7 will be equal to zero. For the same reason, we added the parameter δ to the first expression of the formula 7. In MAUD, the parameters $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are empirically determined. Methods giving more reliable results exist e.g., a nonlinear interpolation technique [Ney 94]. In our method, an empirical definition of parameters is sufficient because the used language model is not entirely stochastic: it utilizes some formal grammatical rules and generate more than one solution [Smaili 91]. Our language model is built up on a corpora of about 400 000 words.

II.3 The Perplexity: a Measure of Language Model

In order to compare two language models, we define first what a good language model is. In speech recognition, a language model is supposed to predict words and to reject bad hypothesis. A language model is better than another if it predicts better and selects better. In order to measure the predictive power of the language model, the speech community uses the inverse of the geometric mean of the probabilities assigned by the model to a large sample of text. This measure is called the *perplexity* (PP). Two kinds of perplexity can be calculated: a corpus perplexity and a test perplexity, in other terms a training perplexity and a test data perplexity like in [Ney 94]. For the training perplexity, we can use the quantity Q of the formula (6) to express the perplexity as follows:

$$PP = \frac{1}{\sqrt[n]{Q}} \quad (8)$$

In order to maximize the likelihood probability of the formula (1), we can obtain the same result by minimizing the formula (8).

II.4 An Automatic Classification

It is very difficult to consider a new classification for each new application domain. Therefore the best way to adapt the language model to a new application is to learn automatically the classes which are necessary to the language model. The problem of looking for a classification in the space of possible classifications is a combinatorial problem. Thus the practical way to approach this hard problem is to design approximated solution algorithms. Thus we chose to use an optimization algorithm such as simulated annealing. In our approach, the transformation function which consists in moving one word from a class to another is not done randomly, contrary to other use of this technique in word classification [Jardino 93][Moisa 95]. As a matter of fact, we decided to supervise the algorithm in the choice of its transitions. Before going further, let us remember the most important principles of simulated annealing.

II.4.1 Principle of Simulated Annealing (SA)

The concept of SA is based on a strong analogy between the physical annealing process of solids and the problem of solving large combinatorial optimization problems. In condensed matter physics, people are interested in obtaining low energy states of a solid: how to arrange the billions of particles in order to achieve a highly structured lattice with a low energy of the system. Metropolis introduced an algorithm to simulate the evolution of a solid in a heath bath to thermal equilibrium [Lutton 86]. Let us give this algorithm before using it in word classification.

1 Start with a high temperature T
2 With a temperature T and until the equilibrium is reached, do
2.1 From the current temperature T of the system and from the current state i of the solid with the Energy E_i, make a perturbation which transforms the state i into state j. The energy of the state j is E_j
2.2 if E_j-$E_i \leq 0$ then state j is accepted as the current state; otherwise the state j is accepted with a probability :

$$P = \min\left(1, \exp\left(\frac{E_i - E_j}{T}\right)\right)$$

3 Change the temperature and goto step 2 until the low (final) temperature is reached

This algorithm permits to the simulation process to be released from a track of a localminimum by doing some transitions with higher energy. This algorithm will converge to one globally optimal configuration with a lower energy. The time spent at each temperature is called *annealing schedule.*

II.4.2 Lexical Classification by Simulated Annealing

The lexical classification can be viewed as combinatorial problem. From a set of N words, how to arrange them on classes in order to obtain a good language model (i.e., a low perplexity)? Thus the parameters of SA have to be adapted to this problem.

The temperature in automatic word classification will play the role of a control parameter. In our experiments, the initial and final temperatures are empirically chosen. The results of these experiments suggest that it is useless to utilize a very high initial temperature (Fig 1.b). As a matter of fact, this high value makes the system explore solutions which are very far from the optimal one. In addition these solutions will be rejected by the algorithm. Figure1.b shows that starting with an initial temperature of 30, the system explores solutions which are very far from the best one. The system gives the same perplexity (about 18) for a long time and when it reaches a temperature of 0.01, the classification will begin to really explore good solutions. For weak initial temperatures, the system gives a very bad classification (see the left curve of Figure 1.b). That is why we set the initial temperature to 0.03 in our experiments. For the final temperature, Figure 1.a shows that the classification converges at a temperature of 1e-5.

To carefully decrease the temperature we choose the function $T_i = T_{i-1} \times \delta$ with $\delta = 0.93$.

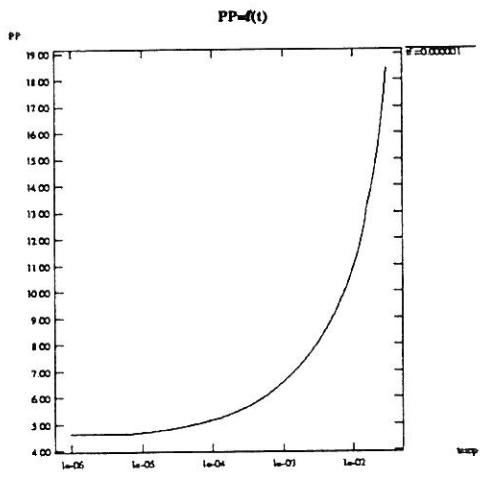

Fig 1.a The evolution of perplexity with a very low final temperatu-

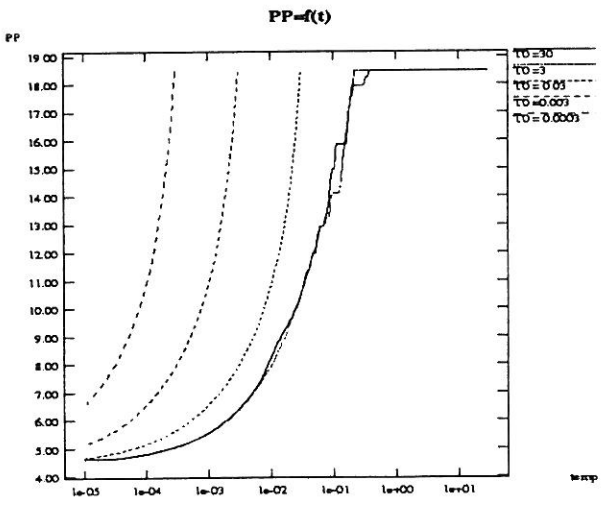

Fig 1.b The evolution of the perplexity with different initial temperatures

The initial configuration of the system will be represented by any initial classification. In our experiments, we tried three initial configurations: all words in the same class, the same

number of words in each class and the randomely merged words (see Fig 2).

The stop criterion is determined by the final temperature of the simulating process.

The energy to minimize is expressed by the perplexity of the language model. In order to make the computations easier and faster, we rewrite from the equation (8) (the formula of the perplexity) as follow:

$$PP = \left(\prod_{i=1}^{n} \frac{N(w_i)}{N(C_i)} \times \frac{N(C_{i-1}C_i)}{N(C_{i-1})} \right)^{\frac{-1}{n}} \quad (9)$$

$N(x)$ is the number of occurrences of x. By taking the logarithm of the formula (9), we obtain:

$$PP = e^{\frac{\sum_{i=1}^{n} \log \frac{N(w_i)}{N(C_i)} \times \frac{N(C_{i-1}C_i)}{N(C_{i-1})}}{n}} \quad (10)$$

By developing the numerator (S) of the right term of (10), we obtain:

$$S = -\sum_{i=1}^{n} \log N(w_i) - \sum_{i=1}^{n} \log N(C_{i-1}C_i) + \left(\sum_{i=1}^{n} \log N(C_i) + \sum_{i=1}^{n} \log N(C_{i-1}) \right) \quad (11)$$

By assuming that the two terms between brackets in the last formula are approximately equal and by developing the sum symbols, we obtain:

$$S = -\sum_{i=1}^{V} N(w_i) \times \mathrm{Log}N(w_i) - \sum_{i=1}^{B} N(C_{i-1}C_i) \times \log N(C_{i-1}C_i) + 2 \times \sum_{i=1}^{V} N(w_i) \times \log N(C_i) \quad (12)$$

V is the number of different words in the vocabulary and B is the number of bigrams. By dropping terms independent of classification, the energy function (E) to minimize is given by:

$$E = -\sum_{i=1}^{B} N(C_{i-1}C_i) \times \log N(C_{i-1}C_i) + 2 \times \sum_{i=1}^{V} N(w_i) \times \log N(C_i) \quad (13)$$

II.4.3 The Improved Simulated Annealing Classification Algorithm

In our experiments, we implemented the classical method of simulated annealing by using a random choice of a target class and a movable word. In this method the annealing schedule is fixed by using this random process $2 \times N$ times (N is the number of different words of the corpora). For this algorithm, we experimented several initial classifications (see Fig 2) by fixing the number of required classes (in this case the number of classes is 139). This number denotes the number of classes generated by our approach on a small corpus of about 2000 words. Figure 2 shows that when all the words are kept in the same class the perplexity decreases from a very high value to a stage nearly equal to 11. In the other experiments, the perplexity decreases from about 10 to about 6. Other experiments show that it is not possible to pass beyond this step. All these experiments exhibit that despite of the low perplexity obtained, the generated classification is not satisfactory because the obtained classes are not well formed.

In our approach, in order to palliate these problems, we propose a simulated annealing algorithm driven by a knowledge based module. The main idea consists in defining a transition function instead of a random transition as in classical approaches. Our transition function specifies the way the classes may evolve during the simulating annealing process. This function permits to make only the plausible linguistic transitions. As a matter of fact, we assume that a word w is moved from one class Cx to another Cy if: $P(Ct_i w) = P(Ct_i w_k)\ (\forall w_k \in C_y)$ *(14)* where Ct_i denotes a context of the word w. The main idea of equation (14) is that two words could be in the same class if they appear in the same contexts. This is inspired from what we did in hand classification. To make that possible in term of time computation, we take into account only the left

context of the word under study. Therefore, if two words appear in the same left context with approximately the same probability, we assume that those words could be merged in the same class.

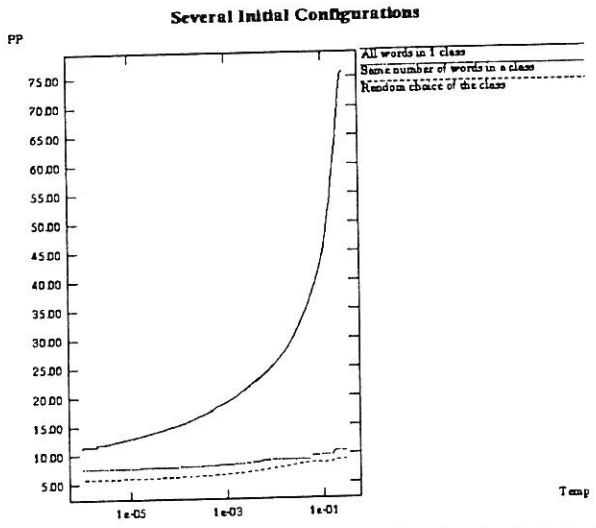

Fig 2 :Tthe evolution of perplexity with several initial configurations by using the classical simulated annealing method

The introduction of the probability in this formula avoids building a lot of classes. As a matter of fact, if we look for the words which have exactly the same contexts, we will probably find a very low number of words which verify this condition and we will have a very high number of classes. A language model with a high number of classes is interesting but its parameters are difficult to estimate for a speech recognition task.

In our approach the transition function is based upon grouping movable words. The algorithm begins by computing, for each bigram of the corpus, its probability by using the previous classification. The probability of each bigram at time t is computed as following:

$$P^t(w_i/w_{i-1}) = \frac{N(w_i)}{N\left(C_i^{t-1}\right)} \times \frac{N\left(C_{i-1}^{t-1}C_i^{t-1}\right)}{N\left(C_{i-1}^{t-1}\right)} \quad (15)$$

At time $t = t_0$, we assume that the conditional probability is reduced to the frequency of the word under study. A list of bloc words is built up with all the words which give approximately the same probability at time t. Each bloc is made up of words which give the same probability. For instance, if $P^t(w_i/w_{i-1}) \approx P^t(w_k/w_{k-1})$, then w_i and w_k will be in the same bloc. Assume that the number of blocs is k. In each bloc, the words which have the same left context are grouped (j sub-blocs on average). The annealing schedule is assumed by the $k * j$ sub-blocs. In another way, the temperature changes when all the words in the $k * j$ sub-blocs will be considered. The transition function allows only moves from a source class to a target one if these two classes belong to the same sub-bloc. We call these words *movable words*. A new class is created whenever all the movable words of a sub-bloc belong to the same class. As a matter of fact, we assume that if words give the same probability in the same contexts and are issuing from the same class, they will probably form an interesting class. A class is removed when it becomes empty. Our approach is illustrated by the algorithm of Figure 3.

III Experimental Results

The improved simulated annealing has been tested on a small corpus of 2000 words made up of reports of an inspector in a nuclear power station and on a corpus (C1) collected from a daily newspaper made up of 40 000 words. Results obtained from this last corpus are illustrated

in figure 4. Results obtained from the small corpus are very interesting: we obtain a low perplexity in a more or less fast time, when compared to the classical method, and the classes obtained are well formed (significant classes). As a matter of fact, some words were arranged into syntactic classes (infinitive verbs, present 3rd person, nouns,...) and others into semantic one (color, direction, moving verbs,...). To really compare the two approaches, we decided to start

```
For Temp = T to ε step 0.93
        compute_energy(E)
        Determination_of_annealing_schedule
        For i= 1 to k /* number of blocs */
                For l = 1 to j /* j the number of sub-bloc */
                        For each movable word
                        begin
                                Make_Transition
                                compute_perplexity(E_t)
                                if E> E_t
                                then
                                        Keep the new classification
                                else
                                        p = random(0,1)
                                                    E - E_t
                                        if p <=   e^(-------)
                                                     Temp

                                then E = E_t and keep the new classification
                                endif
                                endif
                        end
                Next l
        Next i
Next Temp
```

Fig 3 The improved simulated annealing algorithm

the two algorithms with the same initial configuration and obviously with setting the same number of classes (see Figure 5). Two comments can be made: first, the classical simulated annealing is slower than the proposed algorithm and despite the low perplexity obtained in all the experiments, the classes obtained are not well formed. It is true that, in stochastic language model of speech recognition process, we do not necessarly need classes which are syntactically or semantically well formed, because the most important point in this kind of model is to reduce the perplexity. But, typically, it is more advisable to be able to identify at least some classes.

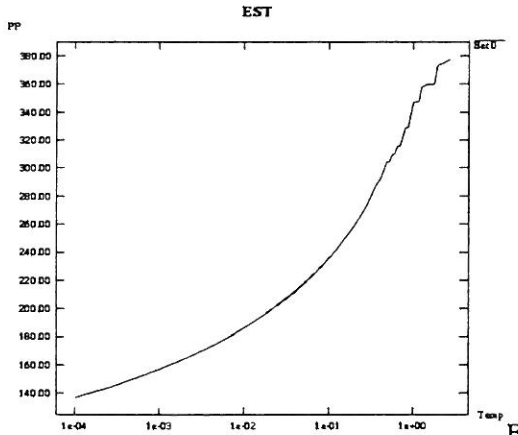

Fig 4 The evolution of perplexity on C1

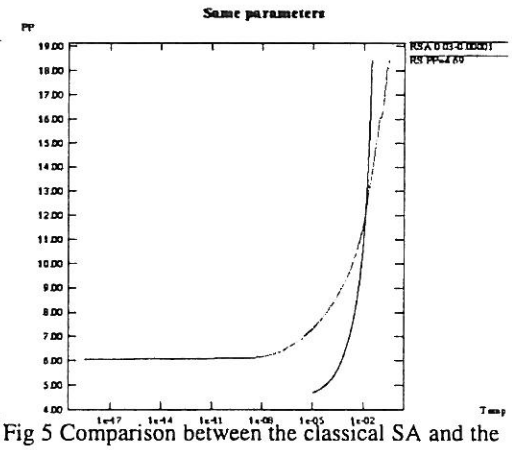

Fig 5 Comparison between the classical SA and the improved SA

This is important when the speech recognition system handles over more the stochastic language model, one which is based on a formal specification of the structures allowable in the language. In this case we have to write rules which take the features of a class word into account.

Second, the transition function permits to better explore the solution space. In the exper-

iment of the Figure 5, we set the stop criterion of classical simulated annealing approach to 4.69 (the convergence value of our algorithm). We stopped the classical SA program at a temperature equal 1e-45 and despite that, the perplexity can not pass beyond the step of 6,71!

IV Conclusion

In this paper, we have presented a new approach for word classification based on a simulated annealing technique. In this approach, we cover up the problem of the random choice of the word to move by introducing some knowledge. From a particular configuration, only some words could be moved from one class to another, by taking into account their contexts and the probability of appearence in these contexts. At each step, the algorithm takes better advantage of the previous classification than the classical SA approach. Our algorithm has been designed to improve the classical SA and to allow to determine the best number of classes to build up for a corpus. As a matter of fact, in the classical methods the number of classes is set and the choice of the number of classes is determined *a priori*. Our experiments show that the perplexity is not a good measure of the language model. For the two low perplexities given by the two approaches, the two obtained classifications are very different. Unfortunately, we do not have a better language model measure today. To really evaluate the two approaches, we have to test the obtained language model on a real speech recognition task. The classical SA has been shown to be computationnally expensive, when compared to ours: with the same number of classes and the same initial configuration and for the same final perplexity which can be reached by the classical SA, our approach is 2.5 times faster. The next step, will be to test this algorithm on a very large corpus and to compute the test corpus perplexity.

References

[Gross 75] M. Gross «Méthodes en syntaxe régime des constructions complétives», Herman, 1975

[Jardino 93] : M. Jardino, G. Adda «Language modelling for CSR of large corpus using auto matic classification of words», 3rd European conference on speech communication and tech nology, Vol 2, pp 1191-1194, Berlin 1993.

[Lutton 86] : J.L. Lutton, E. Bonomi «Simulated Annealing Algorithm for the Minimum Weighted Perfect Euclidian Matching Problem», RAIRO, Vol 20, N 3, pp 177-197, 1986.

[Ney 94] : H. Ney, U. Essen, R. Kneser «On structuring probabilistic dependences in stochastic language modelling», in Computer speech and language, Vol 8, pp1-38, 1994.

[Smaïli 91] : K. Smaïli, F. Charpillet, J-M. Pierrel, J-P. Haton «A continuous speech recognition approach for the design of a dictation machine», 2nd European Conference on Speech Technology, Vol 2, pp953-956, Genova 1991.

[Moisa 95] L. Moisa, E. Giachin "Automatic clustering of words for probabilistic language models", 4th European Conference on Speech Communicationand Technology, Vol 2, pp1249-1252, Madrid 1995.